

Отчет по лабораторной работе №7

Дисциплина: Архитектура компьютера

Клименко Алена Сергеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация переходов в NASM	8
4.2	Изучение структуры файла листинга	12
4.3	Задания для самостоятельной работы	14
5	Выводы	21
	Список литературы	22

Список иллюстраций

Список таблиц

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов:

- условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия.
- безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.


```
mc [klimenkoalena@fedora]:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mc lab7-1.asm [-M--] 5 L:[ 1+19 20/ 21]
#include 'in_out.asm'
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1
call sprintLF
_label2:
mov eax, msg2
call sprintLF
_label3:
mov eax, msg3
call sprintLF
_end:
call quit
```

ботать jmp (рис. ??) Со-
здаю исполняемый файл и запускаю его. Результат совпадает с тем, что находится

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mc
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

в лабораторной работе (рис. ??)

После изменения кода в файле запускаю его (рис. ??)

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Далее меняю код, чтобы он выводил то, что указано в задании (рис. ??)

```
%include 'in_out.asm'
```

SECTION .data

msg1: **DB** 'Сообщение № 1',0

msg2: **DB** 'Сообщение № 2',0

msg3: **DB** 'Сообщение № 3',0

SECTION .text

GLOBAL _start

_start:

jmp _label3

_label1:

mov **eax**, msg1

call sprintf

jmp _end

_label2:

mov **eax**, msg2

call sprintf

jmp _label1

_label3:

mov **eax**, msg3

call sprintf

jmp _label2

_end:

call quit

```
mc [klimenkoalena@fedora]:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
lab7-1.asm [----] 9 L: [ 1+26 27/
%include 'in_out.asm'

SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0

SECTION .text
GLOBAL _start
_start:

jmp _label3

_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

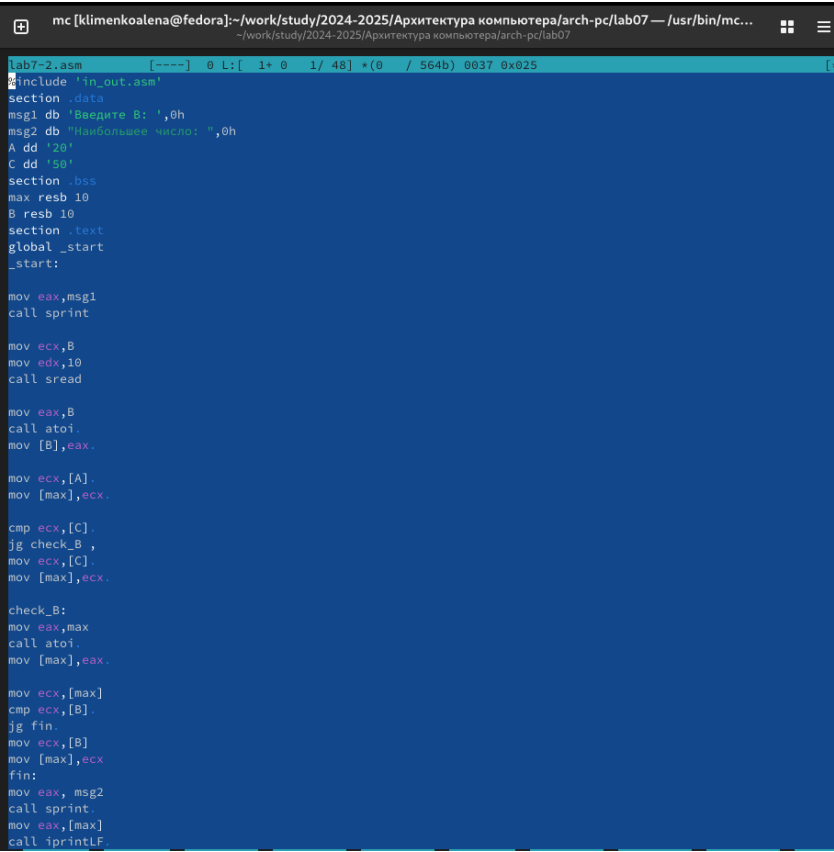
_end:
call quit
```

Проверяю коррект-

ность написания и вывода (рис. ??)

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Создаю файл lab7-2.asm и после изучения листинга 7.3 ввожу код в файл (рис.



```
lab7-2.asm [----] 0 L: 1+ 0 1/ 48 * (0 / 564b) 0037 0x025
#include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db 'Наибольшее число: ',0h
A dd '20'
C dd '90'
section .bss
max resb 10
B resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

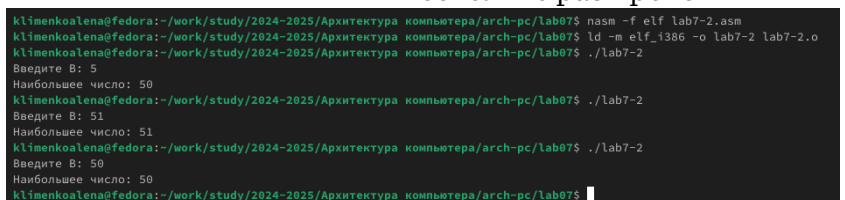
check_B:
mov eax,max
call atoi
mov [max],eax

mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
fin:
mov eax, msg2
call sprint
mov eax,[max]
call iprintLF
```

??)

Несколько раз прове-

ряю корректность работы(рис. ??)



```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 51
Наибольшее число: 51
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 50
Наибольшее число: 50
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

4.2 Изучение структуры файла листинга

Открываю файл листинга с помощью текстового редактора mousepad. (рис. ??) В листинге есть три столбца, не считая первый, в котором просто номер строки. Первый отвечает за адрес строки в файле asm, номер строки указан в шестнадцатеричной системе. Во втором столбце находится машинный код. В третьем находится исходный текст программы. Рассмотрю это на конкретных строках кода, например, 8,9,10 8,9,10 - номер строки в листинге 00000003, 00000006, 00000009 - обозначают на какой строке кода находится команда, последовательность не

постоянна потому что в коде есть пустые строки 803800, 7403, 40 - обозначение команды машинным кодом `cmp... ,0, jz finished, inc eax` - текст программы

```

~ /work/study/2024-2025/Архитектура компьютера/arch-pc/lab07/lab7-2.lst - Mousepad x
Файл Правка Поиск Просмотр Документ Помощь
1                                     %include 'in_out.asm'
2                                     <1> ;----- slen -----
3                                     <1> ; Функция вычисления длины сообщения
4                                     <1> slen:
5 00000000 53                       <1>     push    ebx
6 00000001 89C3                     <1>     mov     ebx, eax
7                                     <1>
8 00000003 803800                   <1> nextchar:
9 00000006 7403                       <1>     cmp     byte [eax], 0
10 00000008 40                       <1>     jz      finished
11 00000009 EBF8                     <1>     inc     eax
12                                     <1>     jmp     nextchar
13                                     <1> finished:
14 0000000B 29D8                     <1>     sub     eax, ebx
15 0000000D 5B                       <1>     pop     ebx
16 0000000E C3                       <1>     ret
17                                     <1>
18                                     <1>
19                                     <1> ;----- sprint -----
20                                     <1> ; Функция печати сообщения
21                                     <1> ; входные данные: mov eax,<message>
22                                     <1> sprint:
23 0000000F 52                       <1>     push    edx
24 00000010 51                       <1>     push    ecx
25 00000011 53                       <1>     push    ebx
26 00000012 50                       <1>     push    eax
27 00000013 E8E8FFFFFF             <1>     call    slen
28                                     <1>
29 00000018 89C2                     <1>     mov     edx, eax
30 0000001A 58                       <1>     pop     eax
31                                     <1>
32 0000001B 89C1                     <1>     mov     ecx, eax
33 0000001D BB01000000             <1>     mov     ebx, 1

```

Убираю один операнд

```

mov eax,msg1
call sprint

mov ecx,B
mov edx,10
call sread

mov eax,B
call atoi
mov [B],;eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B ,
mov ecx,[C]
mov [max],ecx

```

(рис. ??)

В листинге до-

```

19 000000FC E842FFFFFF      call sread
20
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      call atoi
23                          mov [B],;eax
23                          error: invalid combination of opcode and operands
24
25 0000010B 8B0D[35000000]      mov ecx,[A]
26 00000111 890D[00000000]      mov [max],ecx
27
28 00000117 3B0D[30000000]      cmp ecx,[C]

```

бавляется отображение ошибки (рис. ??)

4.3 Задания для самостоятельной работы

Как я поняла я должна решить такой же вариант как и в 6 лабораторной работе, это - 2 вариант Написала программу для нахождения наименьшего числа из

```
klime klime
lab7-- Файл Правка Поиск Просмотр Документ Помощь
klime %include 'in_out.asm'
klime section .data
klime msg1 db "Наименьшее число: ",0h
klime A dd '82'
klime B dd '59'
klime C dd '61'
klime section .bss
klime min resb 10
lab7--
lab7-- section .text
klime global _start
klime _start:
lab7--
klime mov eax,A
klime call atoi
klime mov [A],eax
klime
Наибо mov eax,B
klime call atoi
klime mov [B],eax
Наибо
klime mov eax,C
klime call atoi
klime mov [C],eax
Наиме
klime mov ecx,[A] ; ecx = A
klime mov [min],ecx ; min = A
klime
klime cmp ecx,[C] ; A ? C
Наиме jb check_B ; if A < C |-> check_B
klime mov ecx,[C] ; if A > C |-> ecx = C
klime mov [min],ecx ; min = C
klime
klime check_B:
klime mov ecx,[min] ; ecx = min(A/C)
klime cmp ecx,[B] ; A/C ? B
klime jb fin ; if A/C < B |-> fin
klime mov ecx,[B] ; if A/C > B |-> ecx = B
Наиме mov [min],ecx ; min = B
klime
klime fin:
klime mov eax, msg1 ; eax = msg1
klime call sprint ; вывод
Наиме mov eax,[min] ; eax = min
klime call iprintLF ; вывод
klime call quit
```

3(рис. ??)

```
%include 'in_out.asm'

section .data

msg1 db "Наименьшее число: ",0h

A dd '82'
B dd '59'
C dd '61'

section .bss

min resb 10
```

```

section .text
global _start
_start:

mov eax,A
call atoi
mov [A],eax

mov eax,B
call atoi
mov [B],eax

mov eax,C
call atoi
mov [C],eax

mov ecx,[A] ; ecx = A
mov [min],ecx ; min = A

cmp ecx,[C] ; A ? C
jb check_B ; if A < C -> check_B
mov ecx,[C] ; if A > C -> ecx = C
mov [min],ecx ; min = C

check_B:
mov ecx,[min] ; ecx = min(A/C)
cmp ecx,[B] ; A/C ? B
jb fin ; if A/C < B -> fin

```



```
mov ecx,[B] ; if A/C > B |-> ecx = B
mov [min],ecx ; min = B
```

```
fin:
```

```
mov eax,msg1 ; eax = msg1
call sprint ; вывод
mov eax,[min] ; eax = min
call iprintLF ; вывод
call quit
```

Проверяю корректность работы кода (рис. ??)

```
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
Наименьшее число: 59
klimenkoalena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
```

Написала программу, которая для двух введенных с клавиатуры значение вычис-

```
mc [klimenkoalena@fedora]:~/work/study/2024-2025/Ap...
~/work/study/2024-2025/Ap...
lab7-4.asm [----] 0 L: [ 9+ 8 17/ 55] *(319
SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax.

mov eax, msg2
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp esi, edi
jg var2 ; a > x |-> var2

mov eax, x
call atoi

add eax, -1
mov edi, eax ; edi = aex
jmp fin

var2:
mov eax, a
call atoi
add eax, -1
mov edi, eax ; edi = aex

fin:
mov eax, res ; eax = res
call sprint ; строка
mov eax, edi ; eax = edi
call iprintLF
call quit
```

ляет требуемое значение и выводит результат (рис. ??)

```
%include 'in_out.asm'

SECTION .data
msg1: DB 'Введите значение переменной x: ',0
msg2: DB 'Введите значение переменной a: ',0
res: DB 'Результат: ',0

SECTION .bss
x: RESB 80
```

```

a: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
mov edi,eax

mov eax, msg2
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax,a
call atoi
mov esi,eax

cmp esi,edi
jg var2 ; a > x -> var2

mov eax,x

```

```

call atoi

add eax, -1
mov edi, eax ; edi = aex
jmp fin

var2:
mov eax, a
call atoi
add eax, -1
mov edi, eax ; edi = aex

fin:
mov eax, res ; eax = res
call sprint ; строка
mov eax, edi ; eax = edi
call iprintLF
call quit

```

Проверяю корректность работы кода (рис. ??)

```

klimenkoalena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-4.o
klimenkoalena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-4
klimenkoalena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 6
klimenkoalena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 5
klimenkoalena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$

```

5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходов, а также приобрел навыки написания программ с использованием переходов, познакомился с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7